Constraint Symmetry and Solution Symmetry[*]

David Cohen
Department of Computer Science, Royal Holloway, University of London, UK

Peter Jeavons
Computing Laboratory, University of Oxford, UK

Christopher Jefferson and Karen E. Petrie
School of Computer Science, University of St Andrews, UK

Barbara M. Smith
Cork Constraint Computation Centre, University College Cork, Ireland

Abstract

Symmetry in constraint satisfaction problems (CSPs) has been considered in two fundamentally different ways: as an operation preserving the solutions of a CSP instance, or as an operation preserving the constraints. To reflect these two views, we define *solution symmetry* and *constraint symmetry*. We discuss how these concepts are related and show that some CSP instances have many more solution symmetries than constraint symmetries.

Introduction

Symmetry in a problem can cause difficulties for methods that use complete search to find solutions. Symmetry maps solutions to solutions and non-solutions to non-solutions, so that if the search has established that some partial solution cannot be extended to a complete solution to the problem, the same is true of any symmetric equivalent of that partial solution; unless this is recognised, the search may consider all the symmetric equivalents and prove repeatedly that they do not lead to a solution.

Symmetry breaking, i.e., devising ways of preventing the search from needlessly exploring symmetric equivalents, has become an active research area in constraint programming. Our recent papers (Cohen *et al.* 2005; 2006) stemmed from the surprising observation that in spite of this research

activity, there is no consensus amongst researchers on the fundamental question of how to define symmetry.

Although we have said that symmetry maps solutions to solutions, we intend this as a property of symmetry, not a definition. Nevertheless, some researchers have defined a symmetry of a constraint satisfaction problem as a mapping that transforms solutions into solutions. Others define a symmetry as a mapping that preserves the problem, and as a consequence transforms solutions into solutions. These definitions are distinct; for some problems, they identify different sets of mappings. We show how the two sets of mappings are related and discuss some implications of the definitions.

Definitions

We first fix our terminology by defining a CSP instance.

Definition 1. *A CSP instance is a triple $\langle V, D, C \rangle$ where:*

- *V is a set of* variables*;*
- *D is a universal* domain*, specifying the possible values for those variables;*
- *C is a set of* constraints*. Each constraint $c \in C$ is a pair $c = \langle \sigma, \rho \rangle$ where σ is a list of variables from V, called the constraint* scope*, and ρ is a $|\sigma|$-ary relation over D, called the constraint* relation*.*

An assignment *of values to variables is a set $\{\langle v_1, a_1 \rangle, \langle v_2, a_2 \rangle, ..., \langle v_k, a_k \rangle\}$ where $\{v_1, v_2, ..., v_k\} \subseteq V$ and $a_i \in D$, for all $1 \leq i \leq k$. A* solution *to the CSP instance $\langle V, D, C \rangle$ is a mapping from V into D whose restriction to each constraint scope is in the corresponding constraint relation, i.e., is allowed by the constraint.*

The constraint relation of a constraint c is intended to specify the assignments that are allowed by that constraint. Different domains for different variables can be handled by unary constraints, restricting the universal domain.

We will call a CSP k-ary if the maximum arity of any of its constraints is k. A 2-ary CSP will be called a binary CSP.

In (Cohen *et al.* 2005; 2006), we surveyed definitions of symmetry in constraint satisfaction problems that have appeared in the literature. We adopt the most general view,

that symmetry in CSPs acts on variable-value pairs. Under all definitions of symmetry in CSPs, symmetries map solutions to solutions and non-solutions to non-solutions; they disagree over whether this defines symmetry, so that any bijective mapping on variable-value pairs that preserves the solutions must be a symmetry, or whether it is simply a consequence of leaving the constraints of the problem unchanged. This distinction is critical: the choice we make can affect the symmetries that we find. Below, we give two definitions of symmetry for constraint satisfaction problems that reflect these two views and encompass the types of symmetry allowed by the definitions in the literature.

Note that the essential feature that makes a bijective mapping on a set of objects a symmetry is that it leaves some property of the objects unchanged. It follows that the identity mapping is a symmetry, and the inverse of any symmetry is also a symmetry. Furthermore, given two symmetries we can compose the mappings to obtain another symmetry. Hence, the set of symmetries forms a *group*. The symmetry group that we obtain depends on what property is preserved. Our first definition uses the property of being a solution.

Definition 2. *For any CSP instance $P = \langle V, D, C \rangle$, a solution symmetry of P is a permutation of the set $V \times D$ that preserves the set of solutions to P.*

In other words, a solution symmetry is a bijective mapping, defined on the set of possible variable-value pairs of a CSP, that maps solutions to solutions.

Although we want symmetries to map solutions to solutions, we also want to be able to identify symmetries from the statement of the CSP, rather than by examining its solutions. Our second definition views symmetry as preserving the constraints of the CSP.

To state our definition of a *constraint symmetry* we use the *microstructure* of a CSP instance (Freuder 1991). For a binary CSP instance, this captures the details of the constraints in a graph; we extend it to non-binary constraints and take its complement.

Definition 3. *For any CSP instance $P = \langle V, D, C \rangle$, the mi-crostructure complement of P is a hypergraph with set of vertices $V \times D$. A set of vertices is a hyperedge of the microstructure complement if it represents an assignment disallowed by a constraint, or consists of a pair of incompatible assignments for the same variable.*

The vertices of the microstructure complement are variable-value pairs of the CSP, and a set of vertices $\{\langle v_1, a_1 \rangle, \langle v_2, a_2 \rangle, \dots, \langle v_k, a_k \rangle\}$ is a hyperedge if and only if:

- $\{v_1, v_2, \dots, v_k\}$ is the set of variables in the scope of some constraint, but the constraint disallows the assignment $\{\langle v_1, a_1 \rangle, \langle v_2, a_2 \rangle, \dots, \langle v_k, a_k \rangle\}$; *or*
- $k = 2$, $v_1 = v_2$ and $a_1 \neq a_2$.

Recall that an *automorphism* of a graph or hypergraph is a bijective mapping of the vertices that preserves the edges (and hence also preserves the non-edges).

Definition 4. *For any CSP instance $P = \langle V, D, C \rangle$, a constraint symmetry is an automorphism of the microstructure complement of P (or, equivalently, of the microstructure).*

Example 1. The standard formulation of the n-queens problem as a CSP has n variables corresponding to the rows of the chessboard, say $r_1, r_2, ..., r_n$. The domain of values corresponds to the columns of the chessboard, say $D = \{1, 2, ..., n\}$. The constraints can be expressed as follows, to give a binary CSP:

- for all $i, j, 1 \leq i < j \leq n, r_i \neq r_j$;
- for all $i, j, 1 \leq i < j \leq n, |r_i - r_j| \neq |i - j|$.

Considered as a geometric object, a chessboard has eight symmetries: reflections in the horizontal and vertical axes and the two diagonals, rotations through 90°, 180° and 270°, and the identity. These eight symmetries map a variable-value pair $\langle r_i, k \rangle$ to $\langle r_{n+1-i}, k \rangle$, $\langle r_i, n + 1 - k \rangle$, $\langle r_k, i \rangle$, $\langle r_{n+1-k}, n + 1 - i \rangle$, $\langle r_k, n + 1 - i \rangle$, $\langle r_{n+1-i}, n + 1 - k \rangle$, $\langle r_{n+1-k}, i \rangle$ and $\langle r_i, k \rangle$, respectively.

The CSP formulation hides the symmetry between the rows and the columns of the board, since the variables represent rows and the values represent columns. Placing two queens in the same column, for instance the assignments $\langle r_i, k \rangle$ and $\langle r_j, k \rangle$, is not allowed because of the constraints, whereas placing two queens in the same row, say $\langle r_i, j \rangle$ and $\langle r_i, k \rangle$, is not allowed because a variable cannot be assigned two values. The microstructure complement restores the symmetry between rows and columns by treating equally both reasons for a pair of assignments to be disallowed. Hence, each geometric symmetry of the chessboard is an automorphism of the microstructure complement and so is a constraint symmetry of the n-queens problem, for any n.

Constraint and Solution Symmetry Relationships

Although Definitions 2 and 4 appear to be very different, we state and prove in (Cohen *et al.* 2005; 2006) some simple relationships between solution symmetries and constraint symmetries. The theorems are restated here. First, any constraint symmetry is also a solution symmetry:

Theorem 1. *The group of constraint symmetries of a CSP instance P is a subgroup of the group of solution symmetries of P.*

The converse is not true: a solution symmetry is not always a constraint symmetry. However, the solution symmetry group is also the automorphism group of a hypergraph.

Definition 5. *For any CSP instance P, a k-ary nogood is an assignment to k variables of P that cannot be extended to a solution of P. The k-nogood hypergraph of P is a hypergraph whose set of vertices is $V \times D$ and whose set of edges is the set of all m-ary nogoods for all $m \leq k$.*

Theorem 2. *For any k-ary CSP instance P, the group of all solution symmetries of P is equal to the automorphism group of the k-nogood hypergraph of P.*

The k-nogood hypergraph of a CSP instance has the same vertices as the microstructure complement. For a k-ary CSP (one whose constraints have maximum arity k), the k-ary nogood hypergraph contains the hyperedges of the microstructure complement, and possibly some others. The additional hyperedges represent partial assignments of up

to k variables that are allowed by the constraints on those variables, but do not appear in any solution. The theorem shows, for example, that to find the solution symmetries of a binary CSP instance we need add only the binary and unary nogoods to the microstructure complement and find the automorphisms of the new hypergraph. Note that although local consistency techniques such as arc consistency may find new nogoods in a CSP instance, the full set of k-ary nogoods must be identified from the set of solutions, in general.

Example 2. The 4-queens problem has two solutions, shown in Figure 1. In this case, it is easier to consider the *complement* of the binary nogood hypergraph, in which each edge represents a pair of variable-value assignments that is allowed by the solutions. Figure 1 shows this graph, drawn with each vertex, representing a variable-value pair, in the position on the chessboard of the corresponding square. Each solution is represented as a 4-clique in this graph. Its automorphisms are: the vertices within either clique can be permuted; the vertices in one clique can be swapped with those in the other; the eight isolated vertices (representing unary nogoods) can be permuted; and we can also compose these permutations, giving a group of size $4!^2 \times 2 \times 8!$ (i.e., more than 10^{12}). Since the automorphisms of a graph are the same as the automorphisms of its complement, these are the solution symmetries of 4-queens, by Theorem 2. On the other hand, we have confirmed using NAUTY (McKay 1981) that the constraint symmetry group is just the group of 8 chessboard symmetries.
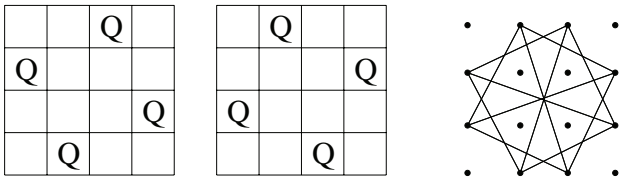

Figure 1: The solutions of the 4-queens problem (left) and the complement of the binary nogood graph (right).

Theorem 1 and Theorem 2 help to clarify the relationship between solution symmetry and constraint symmetry. It is important to distinguish these two kinds of symmetry because, in general, a CSP instance can have many more solution symmetries than constraint symmetries, as we have shown by the example of 4-queens.

Symmetry in CSPs with Few Solutions

If a CSP instance has no solutions, any permutation of the variable-value pairs is a solution symmetry, irrespective of whether the problem has any constraint symmetry. Further, it is easy to see that any CSP instance with very few solutions must have many solution symmetries.

Example 3. Suppose a CSP instance with n variables, and d values for each variable, has only one solution. Any permutation of the n variable-value pairs in that solution is a solution symmetry; so is any permutation of the $n(d - 1)$ variable-value pairs that are not in the solution. The solution symmetry group therefore has $n! \times (n(d - 1))!$ elements.

Example 4. For a CSP instance having 2 solutions, where these solutions have k assignments in common, a solution symmetry must permute the k common assignments amongst themselves, the remaining $(n - k)$ assignments in each solution can also be permuted, and the two solutions can be swapped; the other $(nd - 2n + k)$ assignments can also be permuted. This gives $k! \times (n - k)!^2 \times 2 \times (nd - 2n + k)!$ solution symmetries. The 4-queens problem above is an example, with $k = 0$.

With more solutions, it is more difficult to construct permutations of the variable-value pairs (other than the identity mapping) that preserve the set of solutions, unless there is some inherent symmetry in the problem. Hence, when there are many solutions, the existence of solution symmetries depends on the structure of the solutions, and so on the properties of the CSP. For example, the number of solutions of the n-queens problems increases rapidly with n, and the solution symmetry group appears to consist of exactly the eight symmetries of the constraint symmetry group for $n \geq 7$.

These considerations show that to some extent the existence of solution symmetry reflects how many solutions the CSP instance has, rather than any other property. This can lead to paradoxical results. One way of dealing with (constraint) symmetry is to add symmetry-breaking constraints; the aim is that only one solution in each symmetry equivalence class will satisfy the new constraints. If this can be achieved, the resulting new CSP instance has fewer solutions than the original and no constraint symmetry; but it may have more solution symmetries.

In (Cohen *et al.* 2006), we give an example of a problem (finding a graceful labelling of a certain graph) that has a unique solution, apart from symmetric equivalents. The CSP instance representing the problem has a constraint symmetry group of size 480. Adding constraints to break the symmetry gives a new CSP instance with just one solution; its constraint symmetry group consists of just the identity mapping (all the other symmetry has been broken). However, since the new CSP has only one solution, it has a large solution symmetry group – much larger than the solution symmetry group of the original CSP instance, found using NAUTY. Symmetry breaking has eliminated the constraint symmetry but led to a much larger solution symmetry group.

Symmetry in SAT

One interesting application of these results is to the standard propositional satisfiability problem (SAT); a SAT instance consists of a set of Boolean variables, and a set of clauses in conjunctive normal form on those variables. Each clause is a disjunction of literals, where a literal is either a variable or the negation of a variable. A solution to the SAT instance is a set of assignments to the variables that satisfies the clauses. A SAT instance can be viewed as a CSP instance, in which each clause is a constraint and the length of the clause is the arity of the constraint. The literals of the SAT instance become the variable-value pairs of the CSP instance.

Crawford *et al.* (1996) define a symmetry of a SAT instance as a permutation of the variables that leaves the set of clauses unchanged (where clearly permuting the clauses

or permuting the literals within a clause is not a material change). Aloul *et al.* (2003) allow symmetries to act on literals and hence their definition is similar to our definition of constraint symmetry, applied to SAT. They construct a graph to represent the set of clauses, whose automorphisms are the symmetries of the SAT instance. The graph is similar to the microstructure complement, but has vertices for clauses as well as literals, and colours for different types of vertex.

These definitions correspond to constraint symmetry rather than solution symmetry. However, Theorem 2 shows that the bijections on the literals that preserve the set of solutions are the automorphisms of the k-ary nogood hypergraph; for a SAT instance, the value of k is the number of literals in the longest clause. For instance, to find the solution symmetries of a 3-SAT instance, we would need to find the clauses of length 3 or less implied by the set of solutions. If new clauses can be found without solving the instance, it might be possible to work with a larger symmetry group; Theorem 2 limits the maximum size of clause needed.

Symmetry in Practice

A principal reason to identify symmetry in a CSP instance is to mitigate its effects on search. To prevent the search exploring symmetric equivalent subtrees, two common approaches are to add symmetry-breaking constraints to the CSP, or to use a dynamic method that adds symmetry-breaking constraints when the search backtracks. The dynamic methods have been adapted to deal with the symmetry group rather than the individual symmetries (e.g. Gent, Harvey, & Kelsey 2002), so that large symmetry groups can be defined by a small set of generators and handled efficiently.

In practice, researchers taking these approaches have identified the constraint symmetries of the CSP, rather than the solution symmetries, regardless of their definition of symmetry, because of the difficulty of identifying solution symmetries that are not also constraint symmetries without finding all the solutions. The solution symmetry group is the complete permutation group on the variable-value pairs if and only if the CSP has no solution; hence, finding the solution symmetry group is equivalent to determining whether or not the CSP has a solution. However, since we have shown that the solution symmetries are the automorphisms of the k-ary nogood hypergraph, any additional nogoods that can be found in a CSP instance, for instance by applying local consistency techniques, may allow a symmetry group larger than the constraint symmetry group to be found, and so potentially could reduce the search effort further.

On the other hand, the notion of *interchangeability*, as defined in (Freuder 1991), is explicitly a form of solution symmetry: two values a, b for a variable v are *fully interchangeable* if every solution to the CSP containing the assignment $\langle v, a \rangle$ remains a solution when b is substituted for a, and vice versa. As Freuder notes, in general identifying fully interchangeable values requires finding all solutions to the CSP. He therefore defines local forms of interchangeability; for instance, neighbourhood interchangeability can be identified by inspecting the problem and is a form of constraint symmetry. Nevertheless, subsequent work on interchangeability aims to identify values that are interchangeable in the

set of solutions, and not just in the constraints. For this work, the fact that symmetries form a group is irrelevant; a set of interchangeable values for a variable is dealt with locally, for instance by allowing one of the set to stand for all of them.

Conclusion

We have proposed two definitions of symmetry in the CSP: *constraint symmetry* and *solution symmetry*. The corresponding symmetry groups are the automorphism groups of the microstructure complement and the k-ary nogood hypergraph respectively. Although any constraint symmetry is a solution symmetry, there can be many more solution symmetries than constraint symmetries. Solution symmetry is apparently less useful than constraint symmetry, since finding the solution symmetry group may require knowing all the solutions; however, if k-ary nogoods are discovered, adding the corresponding (hyper)edges to the microstructure complement may allow a larger symmetry group to be found. Most research into symmetry breaking in CSPs implicitly deals with constraint symmetry, we claim. However, a significant strand of research has concerned interchangeability, which is a form of solution symmetry. Our two definitions link these two areas of research into symmetry. We have shown that some definitions of symmetry in SAT are close to our definition of constraint symmetry. Hence, our results are relevant to symmetry research in SAT and could be equally relevant in other areas where symmetry causes difficulties in search.

References

Aloul, F. A.; Ramani, A.; Markov, I. L.; and Sakallah, K. A. 2003. Solving Difficult Instances of Boolean Satisfiability in the Presence of Symmetry. *IEEE Trans. on Computer-Aided Design of Integrated Circuits & Systems* 22:1117–1137.

Cohen, D.; Jeavons, P.; Jefferson, C.; Petrie, K. E.; and Smith, B. M. 2005. Symmetry Definitions for Constraint Programming. In van Beek, P., ed., *Principles and Practice of Constraint Programming - CP 2005*, LNCS 3709, 17–31. Springer.

Cohen, D.; Jeavons, P.; Jefferson, C.; Petrie, K. E.; and Smith, B. M. 2006. Symmetry Definitions for Constraint Programming. *Constraints* 11. (To appear).

Crawford, J.; Ginsberg, M.; Luks, E.; and Roy, A. 1996. Symmetry-Breaking Predicates for Search Problems. In *Proceedings KR'96*, 149–159.

Freuder, E. C. 1991. Eliminating Interchangeable Values in Constraint Satisfaction Problems. In *Proceedings AAAI'91*, volume 1, 227–233.

Gent, I. P.; Harvey, W.; and Kelsey, T. 2002. Groups and Constraints: Symmetry Breaking during Search. In van Hentenryck, P., ed., *Principles and Practice of Constraint Programming - CP 2002*, LNCS 2470, 415–430. Springer.

McKay, B. 1981. Practical Graph Isomorphism. *Congressus Numerantium* 30:45–87. (The software tool NAUTY is available for download from http://cs.anu.edu.au/~bdm/nauty/).